SILICON LABS

Configurable Logic in Microcontrollers Revolutionizes Small Applications

By Stephen To, Applications Engineer, Microcontrollers, Silicon Labs

Introduction

Microcontrollers (MCU) are becoming even more flexible as suppliers integrate more external components on-chip. For engineers, this means a new arsenal of tools and possibilities for embedded development. Configurable logic (CL) hardware is one of these relatively new add-ons to the microcontroller market.

The CL hardware implements combinatorial logic like the basic Boolean AND, OR, or XOR functions, as well as sequential logic like latches. This flexibility allows embedded programmers to easily add their own, custom functionality. From simple signal inverters to more complex Manchester decoders, the configurable logic can operate completely independently from the processor core (CPU). The configurable logic hardware is typically programmed in C and initializes a set of dedicated registers. With C as the programming language, embedded designers will not need to learn another programming language such as VHDL.

Microcontrollers with configurable logic help reduce BOM costs and minimize board size by integrating off-chip glue logic. CPU resources are freed because configurable logic operates independently from the core. In addition, the capabilities of other peripherals are extended when they are connected to configurable logic. With all this flexibility, small, power-efficient MCUs can be used in applications previously exclusive to larger, fully-configurable devices like FPGAs or expensive ASICs.

To understand the capabilities of the highly-flexible features of configurable logic, this paper will explore the general operation of CL hardware on currently available microcontrollers, as well as how configurable logic drives smaller and cheaper applications.

Overview

On a microcontroller, a configurable logic unit (CLU) is a piece of dedicated hardware consisting of user-programmable digital logic that operates without CPU intervention. Multiple CLUs are typically provided on an MCU. Figure 1 shows a simplified block diagram of a single CLU with three inputs and one output.

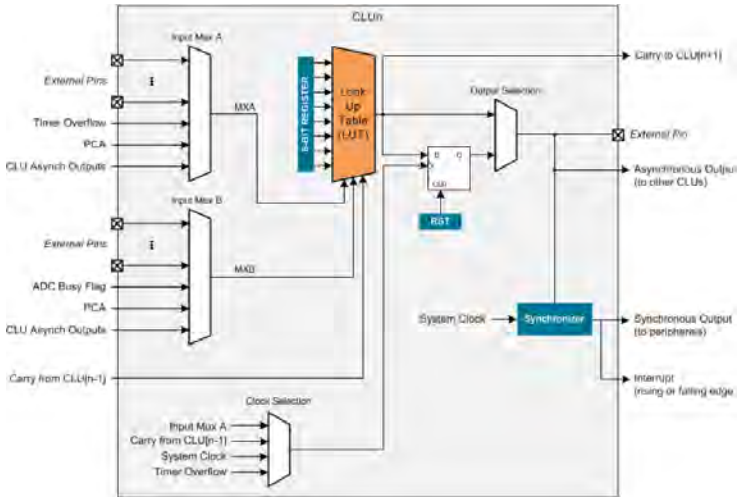

Figure 1: Block Diagram of a Single Configurable Logic Unit

The Configurable Logic Inputs

The three key input signals in a CLU consists of

- Input multiplexer A (MXA)
- Input multiplexer B (MXB)
- Carry signal

Multiplexer A can select from a set of inputs, like external pins or signals from other peripherals. As shown in Figure 2, these inputs can vary from a timer overflow to a PCA channel. The CLU can also operate in conjunction with serial peripherals such as UART or SPI because the external pin inputs can be both a serial peripheral output, as well as a CLU input. Additionally, input multiplexer A can select the output of any CLU, including its own output. Input multiplexer B operates similarly, but with a different set of external pins and signals from other peripherals.

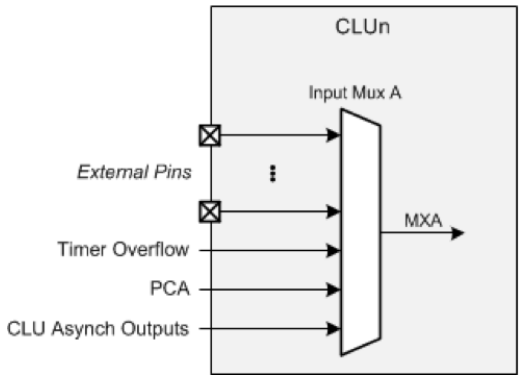

Figure 1: Simplified Block Diagram of a Configurable Logic Unit

The carry signal is the asynchronous output of the previous CLU. This output allows multiple CLUs to be chained and used together. For example, the output of CLU0 is the carry input to CLU1. This signal assignment is fixed. If an embedded programmer wanted to connect the output of CLU0 to CLU3, they would use the destination CLU's input multiplexers A or B.

Look Up Table

At the heart of each CLU is a hardware look up table, which is a multiplexer with MXA, MXB, and the carry as its select signals as shown in Figure 3. This may look familiar as it's similar to a look-up, table-based FPGA block.

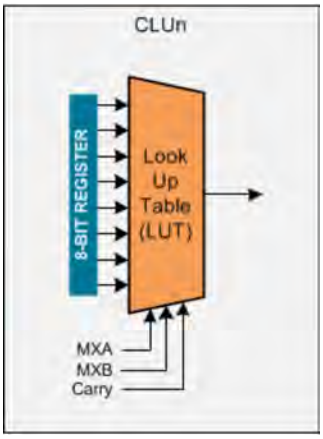

Figure 2: Block Diagram of a Configurable Logic Unit

The embedded programmer chooses the output state for every possible select signal state combination. Since there are three select signals, there are 8 possible inputs, and 8 LUT outputs must be determined. An incomplete lookup table is shown in Table 1.

MXA	MXB	Carry	LUT Output	Comments
0	0	0	TBD	Write to register bit 0
0	0	1	TBD	Write to register bit 1
0	1	0	TBD	Write to register bit 2
0	1	1	TBD	Write to register bit 3
1	0	0	TBD	Write to register bit 4
1	0	1	TBD	Write to register bit 5
1	1	0	TBD	Write to register bit 6
1	1	1	TBD	Write to register bit 7

Table 1: Look Up Table

Implementing AND and XORi

Wth MXA and MXB as the inputs, the Boolean AND could easily be implemented in the CLU. The output is logic high when both multiplexer inputs are logic high, otherwise, the output is logic low. The completed look up table is shown in Table 2.

MXA	MXB	Carry	LUT Output	Comments
0	0	0	0	Write to register bit 0
0	0	1	0	Write to register bit 1
0	1	0	0	Write to register bit 2
0	1	1	0	Write to register bit 3
1	0	0	0	Write to register bit 4
1	0	1	0	Write to register bit 5
1	1	0	1	Write to register bit 6
1	1	1	1	Write to register bit 7

Table 2: Look Up Table – MXA AND MXB

Once the look up table is determined, the desired LUT output is written to a CLU register by firmware during initialization. In the example above, the hex value 0xC0 should be written to the dedicated CLU LUT output register.

As a second example, let's implement a Boolean XOR function with MXA and MXB as the inputs. The output is logic high only when only one of the inputs is logic high; otherwise, the output is logic low. The completed look up table is shown in Table 3, and the value 0x3C should be written to the corresponding CLU register.

MXA	MXB	Carry	LUT Output	Comments
0	0	0	0	Write to register bit 0
0	0	1	0	Write to register bit 1
0	1	0	1	Write to register bit 2
0	1	1	1	Write to register bit 3
1	0	0	1	Write to register bit 4
1	0	1	1	Write to register bit 5
1	1	0	0	Write to register bit 6
1	1	1	0	Write to register bit 7

Table 3: Look Up Table – MXA XOP MXB

The Configurable Logic Output

The asynchronous output of the CLU look up table is connected to the carry input of the next CLU. This asynchronous output can also be routed directly to a port pin, for instance, to connect to and control external components. The output pin of a CLU can also be an input pin for other peripherals.

Synchronizing the Output

Each CLU has an integrated D flip-flop. Asynchronous LUT output can pass through this optional D flip-flop before the signal is routed to a port pin. The D flip-flop can be clocked by number of sources, including the MXA and carry inputs, a timer overflow, or a system clock. The asynchronous output is synchronized to the system clock and routed to other peripherals on the MCU. The rising and/or falling edge of the synchronized output can all trigger an interrupt, allowing firmware to take action upon a CLU event.

Applications and Benefits

Microcontrollers now come with more integrated peripherals and building blocks than ever before, and some of these features have configurable logic that allows complex functions to be implemented without the need for external components. With less than four built-in configurable logic units (CLUs), an embedded programmer now can decide to implement custom and integrated button debouncing, D and SR latching, Manchester encoding/decoding, or Biphase Mark encoding/decoding hardware.

For more information on these examples and start using the configurable logic units, see **Silicon Labs Application Note AN921**, which uses the **EFM8 Laser Bee** (EFM8LB1) microcontroller family featuring 4 CLUs, a 14-bit ADC with automatic sequencing, 4 voltage DACs, calibrated temperature sensor, and a 72 MHz pipelined 8051 core.

® Copyright 2015, Silicon Laboratories Inc. All Rights Reserved. Silicon Laboratories, Silicon Labs, SiLabs and the Silicon Labs logo, CMEMS®, EFM, EFM32, EFR, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZMac®, EZRadio®, EZRadioPRO®, DSPLL®, ISOmodem ®, Precision32®, ProSLIC®, SiPHY®, USBXpress® and others are trademarks or registered trademarks of Silicon Laboratories Inc. All other products or brand names mentioned herein are trademarks of their respective holders.